AFRL-IF-RS-TR-2002-231
Final Technical Report
September 2002

SHARED HUMAN COMPUTER INTERACTION ENVIRONMENT

Honeywell Inc.

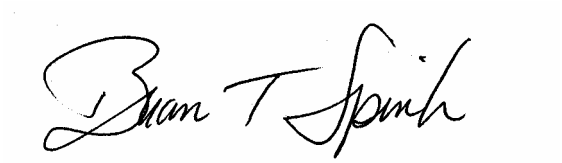
Sponsored by
Defense Advanced Research Projects Agency
DARPA Order No. 8137

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK

This report has been reviewed by the Air Force Research Laboratory, Information Directorate, Public Affairs Office (IFOIPA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

AFRL-IF-RS-TR-2002-231 has been reviewed and is approved for publication.

APPROVED:



BRIAN SPINK
Project Engineer

FOR THE DIRECTOR:

WARREN H. DEBANY, JR., Technical Advisor
Information Grid Division
Information Directorate

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 074-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	September 2002	**Final** **Jun 95 – Mar 99**

4. TITLE AND SUBTITLE

SHARED HUMAN COMPUTER INTERACTION ENVIRONMENT

5. FUNDING NUMBERS

C - F30602-95-C-0177
PE - 62301E
PR - C505
TA - 00
WU - 01

6. AUTHOR(S)

Kyle S. Nelson, Robin R. Penner, Erik S. Steinmetz, and Stephen D. Whitlow

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Honeywell Inc.
3600 Technology Drive
Minneapolis MN 55418

8. PERFORMING ORGANIZATION REPORT NUMBER

N/A

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Defense Advanced Research Projects Agency
3701 North Fairfax Drive
Arlington VA 22203-1714

AFRL/IFGB
525 Brooks Road
Rome NY 13441-4505

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

AFRL-IF-RS-TR-2002-231

11. SUPPLEMENTARY NOTES

AFRL Project Engineer: Brian Spink/IFGB/(3135) 330-7596/Brian.Spink@rl.af.mil

12a. DISTRIBUTION / AVAILABILITY STATEMENT

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

12b. DISTRIBUTION CODE

13. ABSTRACT ***(Maximum 200 Words)***

The Shared Human Computer Interaction Environment program is described with explored collaboration between humans and software agents operating in information-intensive, dynamic command and control applications. Combat search and rescue (CSAR) was the specific domain within which this research was conducted. Over the course of the three-year project, experts from numerous civilian and military organizations contributed to a detailed understanding of CSAR operations that influenced the vision, design, and implementation of the resulting system, the Search and Rescue Assistant (SARA). Expert review indicated that, when implemented SARA would provide order of magnitude improvements to CSAR operations.

14. SUBJECT TERMS mixed-initiative decision support, software agents, multi-agent architecture, human computer interaction, dynamic interaction generation, command and control, combat search and rescue

15. NUMBER OF PAGES 44

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

Contents

FIGURES

Acknowledgements

The authors wish to thank the personnel of the Joint Services Survival Evasion Resistance and Escape Agency (JSSA) for assisting this program. Their domain expertise, support, and involvement were instrumental to the success of this project. We also wish to acknowledge the support and assistance of Dr. Allen Sears and Mr. Tony Osinski in guiding this project.

Acronyms

ATO	Air Tasking Order
C4I	Command, Control, Communication, Computers, and Intelligence
CENTCOM	United States Central Command
CSAR	Combat Search and Rescue
CSARTF	CSAR Task Force
CISA	C4I Integration Support Activity
COM	Component Object Model
COTS	Commercial Off-The-Shelf
CSEL	Combat Survivor Evader Locator radio
DCW	Digital Chart of the World
DIG	Dynamic Interaction Generation
DISA	Defense Information Services Agency
DMA	Defense Mapping Agency (now known as NIMA)
DoD	Department of Defense
DPMO	Defense Prisoner of War and Missing Personnel Office
GOTS	Government Off-The-Shelf
HCI	Human-Computer Interaction
ICM	Interaction Class Model
JSSA	Joint Services SERE Agency
NIMA	National Imagery and Mapping Agency
OLE	Object Linking and Embedding
RESCAP	Rescue Combat Air Patrol
SAR	Search And Rescue
SARA	Search And Rescue Assistant
SARIR	CSAR Incident Report
SERE	Survival, Evasion, Resistance, and Escape
UI	User Interface
USMTF	United States Message Text Format

1. Executive Summary

The 3-year Shared HCI Environment program explored collaboration between humans and software agents operating in information-intensive, dynamic command and control applications. The project was divided into three separate phases, Baseline, Intermediate, and Final. Each phase was designed to answer specific research questions and to build upon the results of previous phases. Each of the three phases consisted of five sequential subphases: Information Gathering, Scenario Analysis, Architecture Definition, Prototype Development, and User-Focused Evaluation.

Several application domains, (including cardiac care, shipboard fire fighting, and oil refinery operations), were considered before deciding upon combat search and rescue (CSAR). Since the collaboration technology needs were similar in each of the candidate domains, CSAR domain was selected because of its relevance to DARPA, the existence of a supportive user community, and the availability of scenarios for system development. Over the course of the project, experts from numerous civilian and military organizations contributed to a detailed understanding of CSAR operations that influenced the vision, design, and implementation of the resulting Search and Rescue Assistant (SARA) system.

A multi-agent architecture, called the Interaction Society (ISociety), was designed for SARA that embodies the following capabilities:

- Automatically retrieve and integrate information relevant to the task at hand, from a broad spectrum of diverse data sources, without explicit direction from the user.
- Maintain and monitor a model of CSAR tasks that describe what the human and automated systems are doing, should do, and can do.
- Dynamically create context-sensitive data visualization displays that reflect the current set of goals, ongoing tasks, required and available assets, and incoming information.

In an ISociety, each agent contributes a specialized skill, and, in return, relies on other society members to provide information within their specialized spheres of interest. Expertise embodied by ISociety agents falls into the following four broad categories. First, interaction agents interact with and help the user (physically and mentally) collaborate with other agents in the society. Second, domain agents track and perform the tasks necessary to support the CSAR domain processes. Third, information agents access and interpret external information sources. Finally, dispatcher agents facilitate the management of individual and group needs as well as the management of the communication between agents.

Three key technical challenges to implementing the ISociety were identified. First, task assistance is required to flexibly divide responsibilities and actions between the computer and human users. Second, the multi-agent architecture must allow agents to operate, communicate, and collaborate with each other and with humans to successfully complete the objectives of the system. Finally, user presentations (i.e., the UI) must adapt to the constantly changing situation and task environment.

Each challenge was addressed to varying degrees during the program, with dynamic UI generation being the primary emphasis. Two prototype demonstration systems were implemented to explore and test solutions to these challenges. The first, called the Map-based SARA Demonstration (or MapDemo), explored the implementation of the multi-agent architectural concepts required by an ISociety and developed HCI concepts for mixed-initiative collaboration to CSAR experts. The second, called the Interaction Facilitator Demonstration (InfacDemo), implemented a Honeywell-developed technique for dynamic UI generation and served as a testbed for a CSAR ontology (or schema).

The project gained an understanding of human-agent collaboration in a real-world military domain, which led to technical advances in several areas including multi-agent architectures and dynamic UI generation. The conclusions reached during this project fall into three broad categories:

- Lessons Learned: The lessons learned cover a range of technical and programmatic issues that similar projects should consider. Two are particularly noteworthy. First, incorporating the user as an active collaborator, rather than simply a task director, proved to be a powerful and novel approach. Second, conducting the research within the context of a real-world domain was crucial. The realism and depth that these scenarios added to the project were key to establishing requirements for collaboration between humans and agents.
- Operational Development: Expert review indicated that, when implemented, SARA would provide order of magnitude improvements to CSAR operations. To realize these benefits, follow-on work is required to address four areas. First, the infrastructure within which the agents operate must be solidified. Second, a task representation to enable efficient task distribution and completion must be defined and implemented. Third, the ISociety must be scaled to allow connectivity to a wide variety and number of information sources. Finally, automated mission documentation, persistence, redundancy and other robustness issues must be addressed.
- Future Research: The mechanics of two types of collaboration, *implicit* and *explicit*, are particularly intriguing. Implicit collaboration arises from a shared understanding of the state of the world including, for example, rules of discourse, semantics, and syntax for topics and situations. Explicit collaboration, on the other hand, is observable collaboration involving requests, responses, and passing of information. Among the interesting research questions is how to share implicit collaboration knowledge with an agent attempting to join a society and what is the role of the user in facilitating this. This agent does not necessarily share the same knowledge as the other agents, may not know its intended role in that society, and may, in fact, possess knowledge that contradicts the knowledge of other society members. Unless a proper foundation is laid for implicit collaboration, explicit collaboration between these agents is very difficult to achieve.

2. Introduction

For three years, Honeywell Technology Center in Minneapolis has been under contract to

DARPA through the Human Computer Interaction Program. The goal of the Shared HCI Environment project that we performed under this contract was to investigate techniques to improve human-computer interaction with multi-agent software systems, emphasizing those that involve the human as an active collaborator.

In recent years the terms "agent" and "multi-agent" have been used quite broadly. For our purposes, a software agent is a computer software system characterized by situatedness, autonomy, adaptivity, and sociability [Sycara, 1998]. A collection of agents interacting to satisfy the objectives of the user is considered multi-agent system. Agents within such systems can interact in a number of ways, including using economic models to compete for tasks. This research program focused on cooperative collaboration rather than competitive collaboration among agents to satisfy the objectives of the user.

The application domains of interest were those in which the human user must interact with and process information from a large number of information sources. We were most interested in domains in which the importance and relevance of information vary according to the specific situation and task. We selected a single domain, Combat Search and Rescue (CSAR), to demonstrate and test our multi-agent architecture. Examples of other potential domains include medical diagnosis, industrial control, military command and control, and intelligence analysis.

In such domains, the user frequently does not have the necessary knowledge to access and analyze constantly changing information, nor the time to determine which information is most relevant to the current situation. On the other hand, the complexity of these domains prevents software agents from completely automating all (or even most) of the tasks necessary to meet the situation requirements in enough detail to provide specific templates in advance. In addition, much of the useful information may not be accessible to or interpretable by the agents (e.g., natural language messages or notations, hard copy photographs, etc.). This requires the human user to act not only as a supervisor, but also as an assistant to the software agents. Software systems such as these, in which human users collaborate with multi-agent systems, are categorized as "mixed-initiative" systems. The system and user work in a collaborative partnership to adapt to the changing situation. The focus of the research performed for the Shared HCI Environment program was to develop human computer interaction (HCI) techniques to improve this collaboration.

We first describe our approach to this research problem during each of the project's three phases. Next, we describe the two software demonstration systems produced to develop and test our research theories and hypotheses. Finally, we discuss our research conclusions and our recommendations for future work.

3. Approach

The project was divided into three separate phases. Each phase was designed to answer specific research questions and to build upon the results of previous phases. We executed a user-centered, rapid prototyping development process that involved user input at every stage, and performed a user review at the conclusion of each phase. Each of the three phases consisted of the following sequential subphases:

1. **Information Gathering**: Obtained and reviewed progressively more detailed information regarding the selected domain to produce a set of scenarios involving human computer interactions in that domain.
2. **Scenario Analysis**: Determined the functional requirements for a prototype system, and detailed the hardware and software requirements placed upon the system.
3. **Architecture Definition**: Developed the architectural concepts and defined the functional components necessary to satisfy the requirements; evaluated the architecture, including a review of relevant government, commercial, and public domain software in terms of these requirements.
4. **Prototype Development**: Used a rapid development methodology to iterate through design/redesign cycles of system components and to integrate those components together into a prototype demonstration.
5. **User-Focused Evaluation**: Completed the user centered design cycle by validating functional requirements through usability testing and user review. Results provided a starting point for the subsequent phase.

The remainder of this section summarizes the goals of each phase followed by sections that describe overall project. Details of each specific phase are documented in the System Development Document [Nelson et al. 1998] for this program.

3.1 Baseline Phase (June 1995 — September 1996)

The goals of this phase were to:

- Select an application domain that identified an interested and supportive user community that included ample scenarios for testing and system development.
- Gather domain information and perform scenario analysis by iterating with domain experts.
- Develop an early conceptual design for the system that includes user-interface mock-ups. Present this to the application domain experts.
- Identify research issues and bottlenecks to fielding the defined system.

3.2 Intermediate Phase (October 1996 — December 1997)

The goals of this phase were to:

- Continue collaboration with domain experts at conferences and meetings. Incorporate this knowledge into the system design.
- Explore each of the research issues identified during the Baseline phase. Extend the prototype to implement, test, and evaluate selected candidate solutions.
- Define and apply metrics to estimate the utility of an operational system for the selected domain.

3.3 Final Phase (January 1998 — December 1998)

The goals of this phase were to:

- Continue collaboration with domain experts through attendance at conferences and knowledge acquisition meetings.
- Focus research and technical development on the most challenging research issues and solidify the overall system design in the process.
- Document the next steps necessary in developing a system beyond the research prototype stage.

4. Information Gathering

Several application domains, (including cardiac care, shipboard fire fighting, and oil refinery operations), were considered before deciding upon combat search and rescue (CSAR). These domains share such characteristics as numerous independent and dynamic on-line and off-line information sources, a fluid task environment, and a user community skilled in operations, not computers. Since the collaboration technology needs were similar in each of the candidate domains, we selected the CSAR domain because of its relevance to DARPA, the existence of a supportive user community, and the availability of scenarios for system development.

The decision to focus on CSAR was clear after an initial meeting with the Joint Services Survival, Evasion, Resistance, and Escape Agency (JSSA). At the JSSA meeting, CSAR experts were interviewed regarding the tasks involved in CSAR, as well as the needs and profiles of users. From this interaction, we determined that not only was CSAR an excellent match to the project's research objectives, but that the speed, efficiency, and accuracy of CSAR operations could be greatly improved through the application of multi-agent collaboration technology. Therefore, a decision was made to direct the efforts of this program towards the development of a Search and Rescue Assistant (SARA).

Subsequent information gathering activities increased the scope of user participation through continued interviews with domain experts. We also gathered valuable information at presentations and demonstrations at several personnel recovery conferences. Of particular note, SARA was presented as part of a vendor display at the Department of Defense Personnel Recovery Conference at Carlisle Barracks sponsored by the Defense Prisoner of War and Missing Personnel Office (DPMO). During this conference, SARA was repeatedly demonstrated to a broad cross section of civilian and military CSAR experts. Over the course of the project, experts from the Civilian Air Patrol, JSSA, CENTCOM, DPMO, DISA, CISA, Pacific Rescue Coordination Center, United States Coast Guard, and others contributed to a detailed understanding of CSAR operations. Feedback from these experts at subsequent reviews and conferences indicates that this understanding is reflected in the vision, design, and implementation of the SARA system.

4.1 CSAR Domain Introduction

The following section provides a justification for the selection of CSAR as the application domain and a very brief introduction to military CSAR operations. For more detailed information, a list of CSAR-related references is provided in the bibliography (Section 10.2).

CSAR operations are characterized by five distinct tasks. Each of these tasks requires access to disparate information sources (satellite photos, status reports, debriefing sessions, weather data, digital maps, etc.). The five fundamental CSAR tasks are:

- **Report**: Inform relevant personnel that a new CSAR mission is required.
- **Locate**: Identify, with high certainty, the most probable position of the isolated personnel (the evader) and verify his/her identity.
- **Support**: Provide necessary support to the evader to prevent their capture and to facilitate a rescue operation.
- **Recover**: Plan and execute the recovery of the evader.
- **Repatriate**: Return the evader to friendly forces and complete the verification and documentation of the actions taken, their results, and lessons learned for future operations.

The scope of the SARA implementation was limited to supporting Joint Search and Rescue Center (JSRC) staff during the second task, broadly involving location of the evader, to both constrain the work and to demonstrate the benefits of our approach. The non-deterministic nature of CSAR incidents makes the relevant information used in locating an evader extremely situation-specific, dynamic, and difficult to predict in advance. In an ocean search, for example, the water current information is critical, but such data is irrelevant when searching small bodies of water like lakes. Furthermore, information sources are the most complex, disparate, and distributed in this phase.

As with other C4I domains, the information is constantly changing the codeword from yesterday is not the same as today. Yet this legacy information must be retained for operational reasons, not just documentation. Information known to the missing individual, particularly security-related information, is current only up to the time they became isolated. The correct codeword for them might be from last week. Recalling that information can be crucial to verifying the individual's identity.

Another difficulty with implementing automated assistance in this domain is the lack of access to electronic information. Crucial information is often not on-line, or is in an incompatible or non-integrated format. For example, photographs might only be in hard-copy format, or witness reports might arrive in natural language via email, telephone, or fax, requiring human interpretation to ascertain the relevance of this information to ongoing operations.

Not only is the information dynamic and situation-dependent, but the tactics used are also strongly situation-dependent and based on dynamically changing higher-level plans and goals. In combat situations, for example, search must be accomplished by means other

than flying search patterns because the searcher's predictability makes the search craft vulnerable to enemy fire. This alters not only the procedures, but also affects the suitable information sources. In this case, JSRC personnel must rely more heavily on satellite photos and other less intrusive information sources.

Most C4I domains share these features, but, because of their complexity, are difficult to scope for research projects such as this one. CSAR, however, has several unique features that make it suitable for such research. CSAR operations are generally smaller in scope and are also more standardized than many others are. This manageability, combined with representative complexity, makes the CSAR domain and procedures an especially productive one for testing the feasibility and design of multi-agent systems.

5. Scenario Analysis

The variability of CSAR incidents places several key constraints on the design of a collaborative multi-agent system. Since information sources cannot be exhaustively enumerated before (or even during) an incident (e.g., it is not possible to identify witnesses to an incident before it occurs), it is necessary to separate the high-level domain knowledge from the execution details. Thus, a mixture of agents performing tasks at various levels is required. In particular, agents that possess knowledge of the high-level goals, plans, and strategies should not concern themselves with details of completing those tasks. Rather, they should rely on agents skilled in specific tasks (e.g., accessing an information source or calculating the most probable search area) to assist them.

Further, because they will participate to varying extents at different times, these agents must flexibly interact without *a priori* knowledge of each other. For example, an agent that calculates the likely search area based on a parachute sighting might rely on wind speed information supplied by multiple location-dependent sources. Until the location is known, this agent has no reason to interact with agents that can supply the wind speed and may not even know such agents exist.

Finally, the fact that much of the useful information may not be electronically accessible or interpretable requires the system to rely on the user not only as a supervisor, but also as an assistant. As with everything else in the CSAR domain, this division of labor is dynamic. The system and user must form a dynamic collaborative partnership to adapt to the changing situation.

A multi-agent architecture must embody the following capabilities to improve the CSAR process:

- Automatically retrieve and integrate information relevant to the task at hand, from a broad spectrum of diverse data sources, without explicit direction from the human user.
- Maintain and monitor a model of CSAR tasks (standard operating procedures) that describe what the human and automated systems are doing, should do, and can do.
- Dynamically create context-sensitive data visualization displays that reflect the current set of goals, ongoing tasks, required and available assets, and incoming

information.

Two subtasks of the Location task, SAR Incident Report (SARIR) completion and geographic situation awareness, were chosen as examples that highlight many of the capabilities above. The first, SARIR completion, is a time consuming task that must be completed before other location subtasks can begin. The SARIR is a message format found within the United States Message Text Format (USMTF) templates. The SARIR, and USTMF in general, has a very complicated and terse format that requires much experience to accurately complete and interpret. Not only are there over 45 applicable USMTF formats for CSAR operations, but the information required by a particular format varies with the situation. Additionally, there are fairly complex constraints on allowable entries that require a variety of interface widgets. Metrics, based on this task, will be collected to estimate the utility that an operational SARA would provide to CSAR Operators, including the time to complete the task and the accuracy of the report generated by the task.

The second task, geographical situation awareness, was selected to provide a test case for the multi-agent architecture and its ability to provide mixed-initiative task assistance. In particular, techniques for sharing information, specifying requests, handling tasks, and presenting information could all be tested using this scenario.

6. ISociety Architecture Definition

The requirements of the selected SAR domain led to an architecture consisting of a "society" of collaborating agents. In an agent society, each agent is treated as a knowledge source; the assistance function is provided by a number of agents interacting to solve a problem for the user. In the case of SARA, not only do agents solve problems for the user and each other, but the user solves problems for the agent as well.

To facilitate these collaborative partnerships, the society provides a framework, called the Interaction Society (or ISociety) [Penner, 1996], that maintains the needs and goals of the participants. In an ISociety group of collaborative agents, each member contributes a specialized skill, and, in return, relies on other society members to provide information within their specialized spheres of interest.

ISociety is derived from the societal behaviors of socialized agents embodied in Rieken's M System [1997], which in turn owes much to Minsky's [1985] earlier conceptual work. In particular, agents with differing reasoning processes are integrated via rules, object representation networks, scripts, and a blackboard system [Nii, 1986].

Expertise (and knowledge) embodied by member agents in an ISociety architecture falls into four broad categories:

- Interaction Agents that interact with and help the user (physically and mentally) collaborate with other agents in the society,
- Domain Agents that track and perform the tasks necessary to support the CSAR domain processes,
- Information Agents that access and interpret external information sources, and

- Dispatcher Agents that facilitate the management of individual and group needs as well as the management of the communication between agents.

This categorization scheme is similar to Decker's [1997], which consists of interface agents, task agents, and information agents. In addition, we add *dispatcher agents* to enable the collaboration. Dispatcher agents facilitate inter-agent communication by routing requests, translating between ontologies, and allowing agents to cooperate without *a priori* knowledge of each other. Dispatcher agents are considered members of the society, not simply as an interaction substrate.

The different types of agents reflect the roles an agent fulfills in the ISociety. Just as in human societies, ISociety agents may fill one or more roles. For example, an agent providing access to a map database (acting as an information agent) might also display its data and the data of other agents (acting as an interaction agent). The role (or roles) an agent fills is important for identifying its needs and responsibilities within the larger society. This is particularly useful when incorporating legacy systems. A system that accesses a database, for example, is incorporated more easily if its role can be limited. The following sections describe each of the roles within an ISociety in more detail.

6.1 Interaction Agents

The demands on an interaction agent are complex in C4I domains like CSAR. There are many possible users, many possible hardware configurations, and many possible software languages, architectures, and operating systems. In addition, the domain parameters and the tasks that must be performed are fluid, interdependent, and intricate. In a simpler domain, the agent can rely on a simple user interface design to manage the collaboration between software and human agents. In a complex domain, however, the interaction agents must have a more sophisticated basis for reasoning about how to communicate with users, and how to support the rapidly changing, situation-dependent task space.

An emphasis of ISociety is the inclusion of human users who actively collaborate by providing the society with areas of expertise not available on-line. Given the non-deterministic nature of the task and the situation, presenting the relevant actions and knowledge of the society in a cogent way, and allowing the human to interact gracefully, are key factors in the overall success of ISociety-based systems. In an ISociety, it is the interaction agents who are responsible for maintaining the user's situation awareness and meeting the society's needs for user interaction [Penner, 1996].

6.2 Domain Agents

Domain agents are versed in the tasks, subtasks, and procedures for a particular application domain. These agents proactively seek to accomplish tasks in their area of expertise. For example, a domain agent versed in the techniques for detecting a CSAR incident, like overdue flights, would contain the knowledge and procedures necessary to detect and confirm a new incident. Domain agents are inherently mixed-initiative as they rely on the society and the user to carry out many of their tasks.

Typically, as with human-human collaboration, it is not possible to have a single agent skilled in all aspects of the application domain. Rather, domain knowledge is distributed

among several interacting agents. An agent responsible for documenting the progress of a mission may not know when such documentation should be completed. This may be the purview of another domain agent that communicates the need for this to the documentation agent.

6.3 Information Agents

Information agents provide information to the ISociety by accessing one or more information sources (e.g., a relational database) and calculating information using data provided by other society members. Information agents are associated with information sources or information needs; for example, an agent that provides weather forecasts might rely on other agents to provide the data necessary for the analysis. By relying on the society for this data, the forecasting agent need only focus on calculations in its own area of expertise.

The user is considered the ultimate information agent in the SARA ISociety. If required information is not available from any other information agent, the user would be requested to provide it. This mixed-initiative approach provides a robust information access capability for situations in which the connectivity, access and location requirements can vary. For example, military CSAR centers can be located in buildings, ships, tents and even airplanes. In each case, the quantity, quality, and access details can vary greatly. A CSAR center based at a large air force base might have direct, on-line access to databases, while a remote CSAR center may only have a voice link. By encapsulating access to this information and providing a transparent means to request data from the user, the participating agents need not be concerned with details of a particular installation.

6.4 Dispatcher Agents

Dispatcher agents address three basic constraints on ISociety implementations:

- The specific information source (or sources) of information might not be known in advance since the situation may alter the availability and relevance of member agents.
- ISociety agents rely on other members of the society to accomplish tasks and provide information.
- The user is tasked with providing information and/or services when other ISociety agents cannot.

Dispatcher agents resolve conflicts between agents and permit member agents to collaborate without the *a priori* knowledge of other ISociety agents. When agents register overlapping interests, for example, the collaboration between agents must either be disambiguated by the individuals involved, or disambiguated by an external decision-maker.

In addition to providing arbitration, dispatcher agents provide essential society services by tracking the current members and their capabilities, knowing how to manage society membership, and managing the high-level goals of the society as an entity distinct from the individuals who participate in it.

7. Prototype Development

Review of the architecture described above identified the following three technical challenges to its implementation:

- Mixed-Initiative Task Assistance: This type of assistance should flexibly divide responsibilities and actions between the computer and human user(s). The resulting system should allow the human to focus on details of the operation or mission rather than details of information search, access, and fusion.
- Multi-Agent Architecture: The architecture should allow agents to operate, communicate, and collaborate with each other and with humans to successfully complete the objectives of the system. Agents within this architecture must be capable of functioning with little or no *a priori* knowledge of other agents.
- Dynamic UI Generation: In a dynamic situation and task environment it is difficult to predict what information will be useful at a given time. Consequently, conventional user interfaces (UIs) lack the robustness and flexibility required to fully support the system. Techniques must be developed to dynamically generate these UIs based on the current situation and ongoing tasks.

To address these challenges, the prototype development was split into two separate systems. Although an integrated demo was considered, the mixture of languages (VB, VC++, and Java) increased the complexity and difficulty of the implementation without shedding any additional light on collaboration issues between agents and humans. The decision to produce two separate demonstrations rather than a single integrated one provided additional opportunity to explore research issues rather than becoming mired in implementation details.

The first demo, called the Map-based SARA Demonstration (or MapDemo), addressed the dual purpose of exploring the implementation of the multi-agent architectural concepts required by an ISociety and presenting HCI concepts for mixed-initiative collaboration to CSAR experts. The second demo, called the Interaction Facilitator Demonstration (InfacDemo), also served a dual purpose. This demo implemented a technique for dynamic UI generation and also served as testbed for a CSAR ontology (or schema). These demonstration systems are described in the following sections.

7.1 Map-based SARA Demonstration (MapDemo)

The MapDemo is a society of agents that fulfill each of the agent roles described in section 6. The agents operate as independent executables, communicating with an independently running dispatcher agent called the Dispatcher. The agents are written in Microsoft's Visual C++ (VC++) and communicate using Microsoft's Component Object Model (COM) via a straightforward OLE Automation interface. These agents, shown in Figure 1, cooperate to access and display global map information to the user.

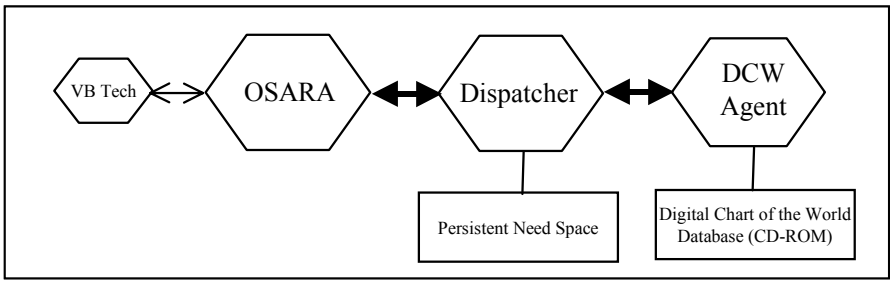


Figure 1: MapDemo Architecture

In the MapDemo, the interaction role is fulfilled by Original SARA (OSARA) which is an encapsulated version of the proof-of-concept demonstration developed during the baseline phase. The emphasis of this early demonstration was on providing a user-centered view of the ISociety and demonstrating other system utilities of SARA. Consequently, the UI for this demonstration was quite advanced. By encapsulating the original system as an agent, the MapDemo capitalized on the benefits of ISociety's inclusiveness by reusing a large portion of the original's UI.

The DCW agent is an information agent responsible for accessing the Defense Mapping Agency's (DMA) Digital Chart of the World (DCW) database, a comprehensive database of global map information. The DCW Agent fulfills society requests for map information including gazetteer information and geographic data.

Note that an earlier version of MapDemo included a Mission agent that fulfilled the domain agent role. This agent contained procedural knowledge necessary to complete the SARIR for a new CSAR mission. During the final phase, the Mission agent functionality was incorporated into the InfacDemo, and will be described in this report as part of that discussion in section 7.2.

The OSARA demonstration serves as a means to test inter-agent communication and communication between agents and humans. In particular, not only does the user task the agents (e.g., by selecting the map location and overlays to display), but in some situations tasks are requested by an agent that must ultimately be completed by the user (e.g., changing a CD in the CD-ROM drive). After describing the major MapDemo components, a scenario is described to illustrate this interaction.

7.1.1 OSARA Agent

OSARA is a multi-process agent implemented using a combination of VC++ and Visual Basic (VB). The University of Michigan's C++ implementation of the Procedural Reasoning System (UM-PRS) [Huber et al., 1976] is used to handle multi-processing during communications with external agents. The UI functions are provided by a VB application with which the VC++ portion communicates.

OSARA incorporates a screen clutter reduction algorithm that attempts to reduce the displayed information to a level that the user can comprehend. Rather than evaluating the actual volume of information that must be displayed, this algorithm determines the number of layers that have been selected and removes those that are less relevant for the

task of perusing a map at a given zoom level. This works well for populated areas (e.g., Europe) but not very well for remote areas (e.g., Newfoundland, Canada) because the volume of information per layer varies between areas. Finally, SARA's choices can be overridden by the user, if SARA defines a view that is not consistent with the user's task.

7.1.2 DCW Agent

The DCW agent encapsulates a legacy system, VPFVIEW, provided by DMA. The fact that this agent's role is as an information agent, and not an interaction agent, simplified the inclusion of VPFVIEW into SARA. The database manipulation routines were virtually untouched while the user interface portions, specialized for older MS-DOS-based machines, could be excised.

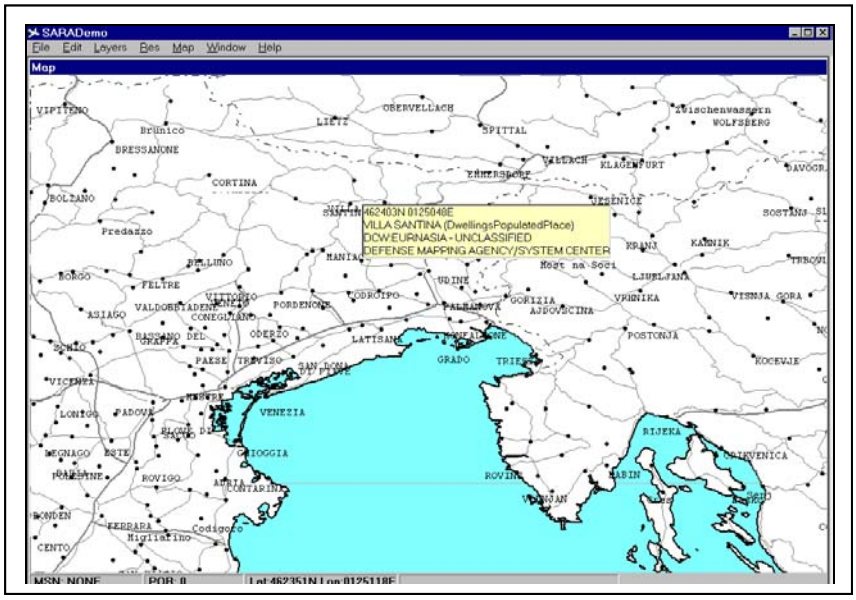


Figure 2: Data supplied by DCW Agent and rendered by OSARA

The DCW database is composed of 4 CD-ROMs each capturing a different portion of the globe. When the agent receives a map request, the DCW agent determines if the proper CD is loaded for the requested grid. If the grid is not supplied, it requests that the original requester supply one. If the CD is incorrect it requests that the society provide a path to the proper CD and suspends the request until the CD is available, moving on to any other pending requests.

7.1.3 Dispatcher

The Dispatcher enables anonymous communication between SARA agents. Every SARA agent, as it starts up, establishes a communication link with the Dispatcher and informs it of the information and services the agent requires from and provides to the society. Whenever an agent has a need that it cannot meet, it asks the society for help by posting its need to the Dispatcher.

The Dispatcher determines the set of agents capable of satisfying the need, selects a

suitable one, and assigns the need to it. It does this through the creation and maintenance of a blackboard data structure called the "need-space", which it uses to track, assign, and monitor the needs of the society as a whole.

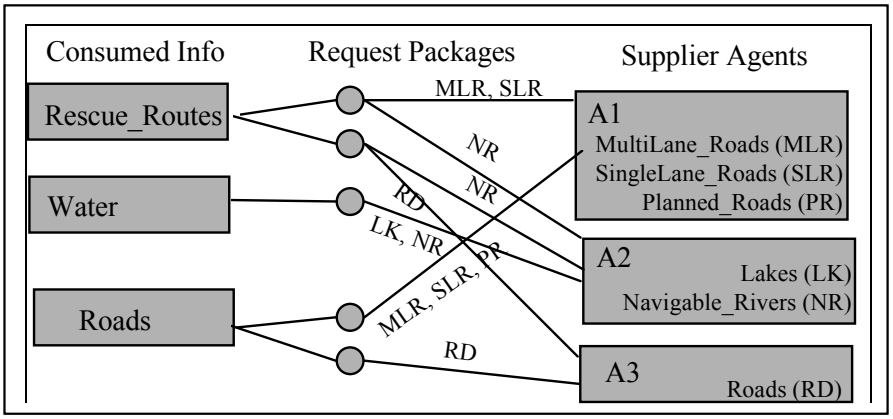


Figure 3: Mappings within the need-space

Via the need-space, the Dispatcher maintains a mapping from each consumed need to the suppliers. This mapping (Figure 3) is contained in a set of *request packages*. Each request package represents an alternative method for satisfying the need. For example, in a society with two different sources of road information, a consumer of road information might have two request packages, one for each of the agent-based map sources. When an agent posts a need to the Dispatcher, the Dispatcher selects one of the possible request packages and assigns the need to the appropriate agents. If the assigned agent fails to meet the need, the Dispatcher selects another suitable request package. Finally, if none of the available request packages meet the need, the Dispatcher assigns the need to the default need supplier (OSARA in our case).

OSARA registers with the Dispatcher as the "default need supplier". Consequently, any need that cannot be satisfied by other SARA agents (e.g., ontological conflicts that are not automatically resolved) will be assigned to OSARA. When assigned a need by the Dispatcher, the user is requested to enter the desired information, or provide a pointer to a file containing that information. The supplied information is then forwarded to the requesting agent by the Dispatcher to complete the need.

In an extension to matchmaking systems [e.g., Cottam et al, 1995], rather than relying on direct agent-to-agent communication once a match is made between requester and provider, the agents continue to communicate directly with the Dispatcher. This not only eliminates the need for a shared ontology among all agents, but also allows agents to cooperate without *a priori* knowledge of other agents in the society. Instead, it lets each agent express the information and services it provides and requires in terms that make sense to it, but not necessarily any other agent in the society. SARA uses the Dispatcher to map between these differences using the need-space.

In practice, many multi-agent systems *do* share an ontology, and requiring the Dispatcher (and user) to rebuild this ontology could be cumbersome. A society working in these

domains would use a Dispatcher pre-encoded with this ontology, essentially a partially instantiated need-space. That is, these limited ontological assumptions are not meant to exclude the use of a shared ontology, but rather to facilitate incorporation of agents that do not share the same ontologies.

For operational use, the current implementation will require many more agents; moreover, those agents will undoubtedly be distributed for practicality and robustness reasons. SARA agents must connect to and communicate with the Dispatcher; but in a large, distributed system this requirement would quickly result in the Dispatcher becoming a bottleneck and a single point of failure. Furthermore, the knowledge needed to locate and connect to a single, possibly remote, Dispatcher could be extensive. Requiring this knowledge of each agent would place unreasonable demands on their implementation, particularly for encapsulated legacy systems.

These observations lead to a planned distributed Dispatcher. With a distributed Dispatcher, ISociety agents always connect to and communicate with a local Dispatcher. That is, every machine with at least one ISociety agent would also have a Dispatcher. Thus, rather than search for a possibly remote Dispatcher, each agent need only check for a local Dispatcher or spawn a new Dispatcher if one does not exist. The potentially complex knowledge and procedures necessary to communicate with distributed agents is encapsulated within the application-independent Dispatcher, making it easily reusable.

7.1.4 Agent Interaction Scenario

To make the interactions among agents clear in the MapDemo, consider the case of the user selecting a geographic location not accessible from the DCW CD currently loaded in the CD-ROM drive. Assume that the CD-ROM drive is a conventional one that requires a person to change the CD, i.e., not a jukebox.

The user selects the new location in the VB portion of OSARA, which causes OSARA to clear the current map and post “needs” for new map information. The information requested consists of the same map data that was present before the location was changed. Each type of map data (e.g., roads, coastlines, etc.) is the subject of a separate need posting.

The Dispatcher receives each of these requests and matches the subject against its Need-space. Assuming that the DCW agent has been registered, these needs map to a request package that can be satisfied by the DCW Agent. In some cases, the need posted by OSARA is a generalization of the information provided by the DCW agent. For example, OSARA requests “roads” while the DCW agent supplies “multi-lane roads” and “single-lane roads”. Thus, the request package for “roads” is relayed to the DCW agent as two separate requests, one for each of the DCW-specific information types.

When the DCW receives each of these requests, it checks to see if the requested geographic grid (supplied with the original need by OSARA) falls (or partially falls) within the extent of the currently loaded CD. If so, it will proceed with collecting and returning the information. In this case, however, the CD is not correct. The DCW Agent will suspend all of the requests that do not match the current CD and post a single need back to the Dispatcher for the proper CD to be loaded which includes the name of the

CD.

Note that the DCW agent does not have knowledge of the agent requesting the information nor does it have knowledge of which agent will satisfy its request to change CDs. Instead, all communication is done through the Dispatcher. In the case of the CD change need, the Dispatcher's need-space does not have a mapping between this need and any supplier agents. In the absence of a successful mapping, the need is assigned to the "default need supplier", which is usually an interaction agent (i.e., the need is eventually handled by the user). If, for example, a separate agent administered a CD jukebox, this "jukebox-agent" would be assigned the need; otherwise the need would be assigned to the default need supplier.

As the default need supplier, OSARA accepts the need and requests that the user switch to the requested CD. At this point, the user types the path to the new CD and informs OSARA that s/he is finished. OSARA then relays the information to the Dispatcher that subsequently passes it on to the DCW Agent.

Once the DCW Agent receives word that the CD has been changed, it reactivates the suspended needs, verifies that the CD is correct, and collects the requested data. As each type of information is collected, the DCW Agent informs the Dispatcher when the data is present and to which file the data was saved. As each part of the request package is completed, the Dispatcher informs OSARA of the location of the data. Finally, OSARA reads the data from the files and renders the map. Note that the information file is a copy and OSARA is free to do whatever it wants with it. Consequently, it is OSARA's responsibility to delete the file when it is finished with it.

7.1.5 Testing Procedures and Results

The MapDemo was tested to evaluate the screen clutter reduction algorithm, verify that requests for DCW data and associated requests for CD changes were correct and consistent, and identify memory leaks and other subtle bugs. The test data included 15 different latitude and longitude locations that were used as the map center and 8 place name locations. These included locations from each of the CDs and some that fell on the border between two or more CDs. The specific locations are found in Appendix 1.

For each location or placename, the location was selected at the map's standard zoom level. Once the map was drawn the map was zoomed out one level, in one level, and back to the original zoom level. This resulted in the map being drawn and data being accessed 5 times for each test. The map layers included oceans, roads, city locations, city names, coastlines, and both administrative and international boundaries.

The testing was effective at eliciting memory leaks and other bugs that occur over periods of long use. The serious problems were fixed while those problems that could be categorized as benign or as enhancements were considered out of the current scope and added to a "wishlist" for subsequent work.

7.2 Interaction Facilitator Demonstration (InfacDemo)

The Java-based InfacDemo, the second demonstration prototype, implements Dynamic Interaction Generation (DIG) technology [Penner & Nelson, 1997] and an object-oriented CSAR schema. DIG is used to dynamically generate a UI to enter SAR Incident Report (SARIR) information for a CSAR Mission. The UI allows the user to create and modify CSAR Schema objects which are then used to generate a SARIR (Figure 4) object instance which can eventually be "submitted" by exporting it as a text file in the USMTF format (Figure 5).

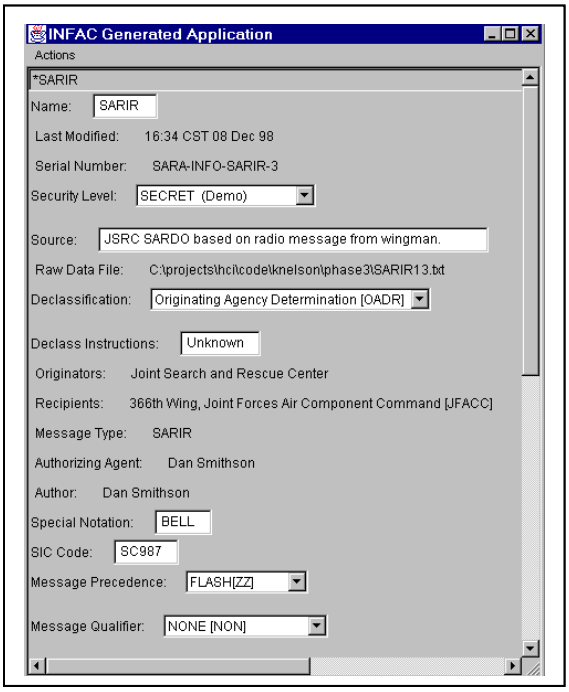


Figure 4: CSAR Schema object rendered by DIG.

The CSAR Schema is a loose combination of the Command and Control Schema (C2 Schema) [Carrico, 1996] that is being constructed to support military-based software systems and a model of the Joint Personnel Recover process described in [Fernandez et al. 1996]. The collection of instantiated schema objects represents the current CSAR situation, entities, and information.

The high-level structure of the CSAR Schema consists of six main branches: Situations, Information, Entities, Domain Primitives, Task Behaviors, and Plan Objects. The latter two of these, Task Behaviors and Plan Objects, are only sketched in and not used in the InfacDemo. Consequently, only the Situation, Information, Entity, and Domain Primitive branches will be described in any detail. The diagrams of the current CSAR Schema can be found in Appendix 2.

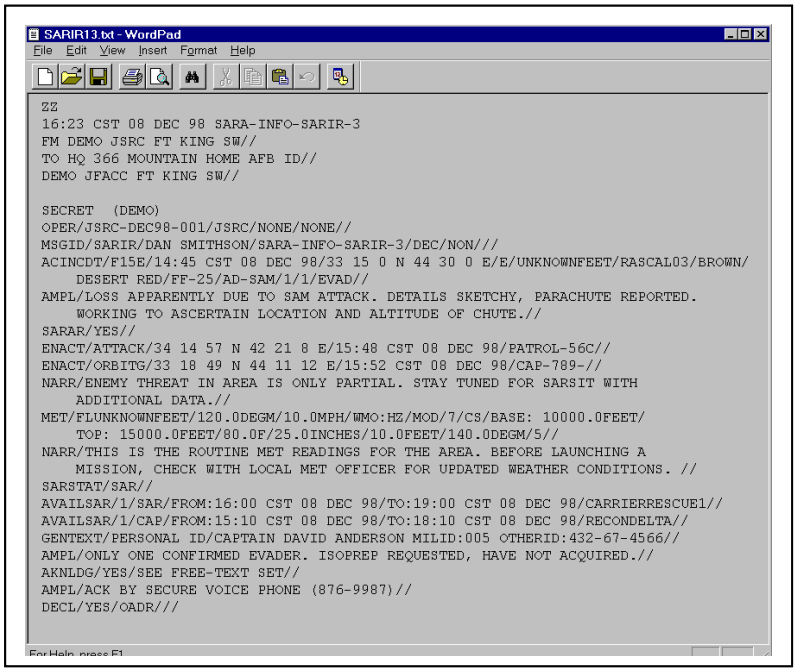


Figure 5: SARIR represented in USMTF format

The main classes of objects in the CSAR schema include:

- Entity Objects represent things in the domain such as materiel, organizations, people, and locations. Entities can act on or participate in the task and situations found in the domain.
- Situation objects represent the situations and roles that involve the entity objects, including objects such as missions, commanders, and pilots. This is separate from the entities that participate in the situation, since an individual entity, such as Major Jones, can fill multiple roles or participate in multiple situations.
- Information Objects reflect the factual data about other application objects. For example, details of a meteorological phenomenon will be based on one or more weather data objects.
- Domain Primitive Objects encapsulate the mapping to native types (e.g. java.lang.String) and other basic data referenced by other CSAR schema objects.

The CSAR Schema will enable much of the necessary work to be done automatically, when the InfacDemo is integrated as one of many agents in SARA. Each agent will share knowledge of the CSAR Schema and perform different aspects of the overall job. Agents

will convey the information and services they provide in terms of the CSAR Schema. This makes inter-agent collaboration significantly easier, and limits the need for ontological synchronization techniques. Information agents that access various databases and other information sources will supply and maintain relevant CSAR Schema instances (e.g., an ATO-Agent may supply information for ongoing missions). Domain agents will provide the higher-level knowledge to filter out and select candidate values from among these instances. For example, suppose the user is selecting assets to form a CSAR Task Force (CSARTF) to rescue an evader. Rather than displaying the list of all assets, a domain agent would screen the list of assets according to the roles that must be filled in the CSARTF (e.g., list only those aircraft capable and available to provide Rescue Combat Air Patrol (RESCAP)).

It is easy to see how the CSAR Schema will facilitate agent integration. By focussing on certain objects and refining their details, the user is conveying information to the agents. Agents use this information to form queries to locate missing information, keep known information current, and provide task assistance based on the user's current focus. For example, consider the aforementioned task of selecting assets to provide RESCAP for a CSARTF. The CSAR Schema object representing the CsarTaskForce, a subclass of CombatMission, consists of, among other things, a RescueCombatAirPatrol submission. The requirements of the RescueCombatAirPatrol (e.g., required capabilities of the aircraft) together with details of the parent CsarTaskForce object (e.g., location, time frame, air threat information, etc.) can be used to form a detailed query specifying the requirements of the aircraft needed. An agent that is tracking available assets, for example, might satisfy this query. These assets would then be presented to the user who could make the final decision.

As another example, an initial SARIR recipient list could be automatically generated by looking at the organizations referenced by other CSAR Schema objects (e.g., the evader's unit, the units of the available CSAR assets, etc.). The user would be able to add and remove items from this list and even create objects that represent organizations that are not currently known to SARA.

Furthermore, much of the information for a particular CSAR incident is unknown, especially in the early stages. The areas of the domain model that coincide with the users current focus (or interest) can be used to automatically generate queries for this unknown information. This same mechanism can be used to summarize the currently known and, more importantly, unknown information for the user. The summary would be maintained as the agents (and humans) provide the unknown information.

7.2.1 Dynamic Interaction Generation (DIG)

(DIG) is a Honeywell Technology Center developed technology that enables inclusion of the human expert as an active collaborator in a multi-agent system. The current implementation (Figure 6) contains an Interaction Facilitator that manipulates an Interaction Class Model (ICM). This ICM supports the display of a UI to create and define CSAR Schema object instances.

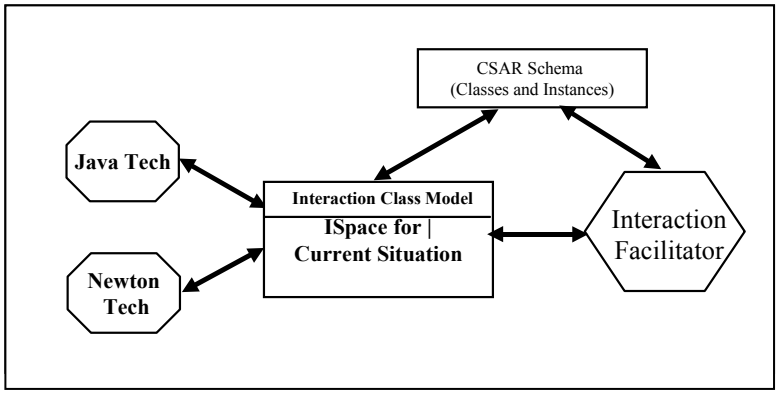


Figure 6: InfacDemo Architecture

The ICM is a description of the interaction *levels* that comprise a user interface to a complex process. It also contains information about the composition of each level in terms of items at the next lower level. These levels are illustrated in the fragment of the ICM that is shown in Figure 7. Each object in the ICM is associated with a real-world object called the object's *referent*. In the case of InfacDemo, the referent is a CSAR Schema instance. As the user changes the focus of the UI, by changing the referent, the ICM object adjusts to the details of the newly selected referent. DIG requires knowledge of only the upper portion of the CSAR Schema and the Domain Primitives. This portion is largely domain independent; thus, DIG can be easily applied to other domains. The DIG-required portion of the schema is outlined in Appendix 3.

At the top level of the ICM is the *UIApplication*, which represents a collection of tasks that are required by a human participating in semi-automated mixed-initiative collaboration (e.g., managing the resources of a building (environment, security, mechanical) or conducting a combat search and rescue operation). Shown in Universal Modeling Language (UML) notation, this top line of the diagram in Figure 7 declares the SARApplication to be a specialization of the more general ObjectApplication, which in turn is a specialization of the more general UIApplication.

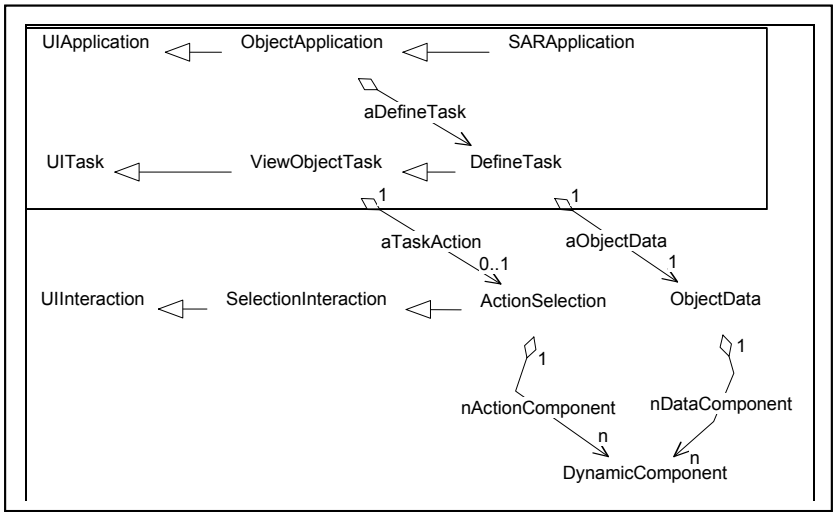


Figure 7: Compositional Levels in the Interaction-Space

An application is composed of various *UI Tasks*, which are roughly analogous to a dialog or application window. Figure 7 shows that the DefineTask is a specialization of a ViewObjectTask, and is one of the sub-elements (or components) of an

ObjectApplication. ViewObjectTask provides the ability to view and interact with the details of a domain object (e.g., a CSAR Schema instance); DefineTask is a specialization of the ViewObjectTask that enables a user to change as well as view the details of the referent, including its properties, children, and associates.

The next level of the ICM is the *Interaction* level. Interactions are roughly analogous to a subframe within a larger window (e.g., a scrolling panel or menu bar). In Figure 7, DefineTask has three Interactions as its sub-components. First, ObjectData is an interaction whereby an object's data is interrogated and modified. Second, ObjectSelection allows the object's children to be selected. Finally, ActionSelection provides the actions that can be taken on the object or the view of the object. The ObjectSelection and ActionSelection are inherited from ViewObjectTask. As shown by the number adornments on the "has-a" links between the task and its interaction, only the ObjectData is required. If there are no actions or child objects associated with the current referent (such as "Submit" might be for a report, or "Enable" might be for a heating system), the ActionSelection interaction is not required.

Interactions are, in turn, made up of *Components*, which are roughly analogous in function and complexity to UI widgets (e.g., value sliders, trend graphs, entry boxes, menu lists). Components are not, however, directly tied to UI widgets; rather they are a device-independent representation of the interactions rendered by those widgets. For example, a city bus schedule is represented in the ICM as an element based on the ItemOverTime class. If it is presented on a display screen at a kiosk at the bus station, it would be represented by a different widget (a schedule timeline) than it would be if it is delivered over a telephone (a series of times and stops, presented sequentially in spoken language).

Despite their eventual mapping to very different user interface widgets, the interaction design required by any bus schedule interaction in the ICM is the same: One which conveys the time-event pairs that make up the schedule. Thus, the same ICM objects represent both user interfaces. In other words, the ICM can fully specify descriptions of the ongoing interaction with the human, without reference to specific hardware. This hardware independence allows the ICM to be mapped to whatever hardware is currently available, most appropriate, or preferred by the user.

The lowest level of the ICM provides additional hardware independence that is not shown in Figure 7. Components are broken down into interaction *Primitives* to allow DIG to separately handle information and interactors. For example, the name of a choice can be separated from the actual choice, contributing to the ability of DIG to compose multiple interfaces from the same information. The same ICM representation would be used to generate labeled buttons ("Print") on a CRT, where the information appears *on* the interactor or keypad telephone interactions ("To Print press 1"), where the information about the action must come *before* the name of the interactor.

An important feature of the ICM is the knowledge it embodies about the *process* of designing interactions that are required to meet task needs [Penner & Nelson, 1997]. The information contained in the *compostionality* of higher levels by lower levels itself defines the general design of the interactions for particular tasks and sub tasks. Making

this compositionality abstract, rather than concrete provides the responsiveness to situations. For example, the Dynamic Component element shown in Figure 7 cannot actually be instantiated; instead, it must be specialized into a more concrete element (like a state setter dynamic component, or a selector dynamic component). Using the situational parameters, like the task in which it occurs or the type of data it represents, a Dynamic Component will determine which of its specializations is most appropriate.

Instances of the ICM are dynamically instantiated, maintained, and destroyed as the user changes focus and the domain model is modified. The set of dynamically varying ICM instances is referred to as the Interaction Space (ISpace). Within the InfacDemo, the Interaction Facilitator manages the ISpace and chooses the specific hardware devices to render the ISpace for the user. It does this through communication with one or more "Interaction Tech" modules. Each Interaction Tech controls a specific interaction device set (e.g., personal digital assistant, keyboard and monitor pair, etc.). The Interaction Facilitator partitions the ISpace among the available Techs.

The InfacDemo implements two Interaction Techs. The JavaTech controls a monitor/keyboard device set running on a Windows NT system with a Java virtual machine. The JavaTech is the primary Tech and was used to generate the screendump used in Figure 4. To test the device independence of the ICM, a NewtonTech was implemented to map the ISpace to a personal digital assistant (a Newton). The ISpace is transmitted to the Newton via a TCP/IP connection and displayed for the user with Newton-based widgets.

7.2.2 Testing Procedures and Results

The majority of the testing was done using the JavaTech to map the ISpace to the screen. Testing consisted of using the DIG-based UI to create and edit objects to complete a SARIR report. A subset of CSAR Schema objects was created in advance to simulate preexisting instances (e.g., those that are provided by other agents). Since this is a demo system, object persistence and change history were not implemented.

The resulting performance is comparable to conventional user interfaces when run on computers with a 200MHz Pentium or faster. Slightly slower performance is seen on a 90MHz Pentium, but the response is well within human tolerances. Analysis showed that while the ISpace modifications happened very quickly, the layout of the resulting Java widgets was a considerable time sink. Efficiency improvements of the layout manager and general improvements to the Java language interpreters should render any performance concerns moot.

One interesting observation was that the modifications to the ISpace following a change to the CSAR Schema instances where proportional to the CSAR Schema change. That is, if a small change was made to the domain objects, the ISpace required only a small modification whereas a larger change required more modifications. For example, changing a primitive (such as the "Message Precedence" in Figure 4) results in an almost imperceptible change in the UI, while creating a new object via one of the Actions results in a larger change.

8. User-Focused Evaluation

At the conclusion of each phase of the project a user evaluation was conducted. These evaluations are summarized here. The SARA Baseline user evaluation was conducted at JSSA headquarters at Fort Belvoir, VA on 20 June 1996. Approximately fifteen people representing JSSA, AFRL, and other DoD organizations were present. The early SARA prototype was demonstrated and feedback and comments were collected.

The Intermediate phase user evaluation was conducted in two parts. The first, held 15 October 1997 at JSSA headquarters, Fort Belvoir, VA, focused on quantifying the utility of SARA's mixed-initiative assistance for CSAR tasks. The second, held 27-30 October 1997 during the Dept. of Defense Personnel Recovery Conference at Carlisle Barracks, PA, collected more general feedback from the users with a focus on future directions for SARA.

Both reviews met their intended objectives. As a result of the JSSA review, SARA is estimated to produce order of magnitude savings in time to complete CSAR tasks for all users with comparable or improved accuracy over current procedures. The accuracy improvements depend on the experience of the CSAR operator. More experienced personnel see comparable accuracy while inexperienced (and, incidentally, more typical) operators see marked improvement in accuracy. Finally, we initiated a “future directions” requirements document with the information gleaned from the Personnel Recovery Community review.

The final SARA user evaluation was conducted in two parts. The first, held 4 August 1998 at the Honeywell Technology Center in Minneapolis, MN, evaluated SARA from the context of users of the ISociety multi-agent architecture. The second was a CSAR expert evaluation conducted by JSSA personnel who have served as domain experts over the course of this project. Results of both reviews will be included in plans for future SARA development.

The ISociety user review proved successful in identifying which issues require careful consideration when implementing SARA and which are straightforward to solve. It also identified more mature areas of research that have addressed similar issues. The results from the CSAR expert user review were encouraging but minimal. The software successfully loaded and ran, but since only minimal task support was implemented, the system had limited operational utility. This is to be expected when the primary focus of the project is research rather than system development. The evaluation succeeded in underscoring the need for a follow-on program specifically targeted to operational development of SARA.

9. Conclusions

A multi-year research project applied to a specific and complex domain, such as this project, results in a broad range of conclusions. These conclusions are grouped into three loosely defined categories:

- Lessons learned that future projects with similar objectives should consider.

- Recommendations for the development of the prototype towards operational use.
- Directions for continued research.

9.1 Lessons Learned

The biggest technical challenges involve developing the instance-level schema knowledge, fusing the resulting information (and knowing when we were done), and developing a rich query language to support the behaviors SARA needs. Fortunately, many of these issues are being addressed to varying extents by other researchers, with some approaches being fairly mature and expected to be available commercially in the near-term.

Developing a complete shared ontology, such as the CSAR Schema, is an expensive and time-consuming process. For this project, using the existing C2 Schema as a starting point made implementation of the CSAR Schema possible within the scope of this project. Consequently, not only should future projects spend time to look for existing schemata when creating a shared ontology for their agents, but work should continue on shared schemas to ensure they evolve into a robust schema generally applicable to military systems.

Incorporating the user as an active collaborator, rather than simply a task director, has proved to be a powerful and novel approach. For example, to make data fusion tractable in the SARA domains, the human will have to play a role in conflict resolution. As "fusion agents" are created, this role may be reduced, but even those agents may reach a point that still requires human intervention. Making this assumption enables significant task support to be provided to the human without becoming bogged down with trying to automate every aspect of the domain (e.g., planning a recovery mission). In complex military domains, this simplification is crucial.

The ontological conflict detection and resolution techniques used by the Dispatcher improved the system's robustness and should be extended. However, the lack of ability of the system to use a shared ontology needs to be rectified. In practice, many multi-agent systems share an ontology and requiring the Dispatcher (and user) to rebuild this ontology would be cumbersome. Instead, the Dispatcher should be pre-encoded with a shared ontology, essentially a partially instantiated need-space. The conflict resolution techniques would then facilitate the incorporation of agents that do not use this shared ontology.

Finally, conducting the research within the context of a real-world domain was crucial. The realism and depth that these scenarios added to the project were key to establishing requirements for collaboration between humans and agents. In addition, the domain should be selected before beginning technical work and all parties should agree. Once selected and technical work has begun, it is important to not change domains. Furthermore, the non-technical aspects of the domain should not be ignored (e.g., interest of the user community).

9.2 Operational Development Recommendations

Due to the project's emphasis on HCI technology development, the resulting SARA demonstrations are not intended for operational use. Projected CSAR operations improvements can only be realized through additional effort to develop an operational system. The following sections describe four general areas that must be addressed.

Infrastructure - Before further development can proceed, the infrastructure within which the SARA agents operate must be solidified. The SARA agents currently function within a fragile infrastructure intended only to support the research prototype. It is not, for example, intended to be distributed across several computers or to support easy addition of new agents. This additional work will apply and/or extend COTS/GOTS technology to provide the infrastructure needed for a robust, solid SARA implementation.

Task Support - A powerful aspect of SARA is the ability for humans and computer agents to flexibly share tasks and cooperate to complete them. The current task representation is limited to only a small portion of the CSAR domain. The existing representation cannot be easily extended to other areas within CSAR, nor can it be adapted to other domains. This additional work will develop a formal representation of tasks and tests this representation. This representation would enable efficient task completion by dynamically assigning responsibilities to one or more human users or intelligent agents.

Connectivity - SARA will require access to a wide variety of information sources, each with its own unique requirements. Some information is stored in relatively static databases, while other information is updated frequently. Some information is synthesized, e.g., the estimated location of a missing pilot might be synthesized from a reported parachute sighting, parachute drag rates, and local wind conditions. Other information is only available from off-line sources and must be supplied by human users, e.g., hard-copy photographs. In addition, SARA must be capable of handling contradictory, redundant, or obsolete information, as well as information stored at multiple security levels. Furthermore, some information sources may not be permanently available, or be temporarily inaccessible. SARA must also be capable of coping in these situations. This additional work will focus on SARA's connectivity to this wide variety of information sources, test the system's scalability, and extend the infrastructure to accommodate additional requirements.

Robustness - To become operationally useful, the progress made in the previous areas will need to be solidified. We must address those issues that are commonly glossed over during rapid prototyping and those items that were out of scope. For example, SARA requires functionality to document CSAR missions and to access that documentation. Furthermore, important task knowledge and information sources that were previously out of scope will need to be added.

9.3 Research Directions

We have recently become interested in the basic mechanics of the collaborative process. It is important to note that there are two types of collaboration, implicit and explicit. Implicit collaboration arises from a shared understanding of the state of the world. This

provides a level of society cohesiveness and knowledge synchronization for collaborating agents by providing implicitly agreed upon rules of discourse, semantics, and syntax for topics and situations. Explicit collaboration, on the other hand, is the collaboration between agents that is observable. It involves requests, responses, and passing of information. What we are beginning to believe is that, unless a proper foundation is laid for implicit collaboration among software agents, explicit collaboration is very difficult to achieve without the explicit sharing of information that should be implicit (like meta-data, language, format, etc.).

Implicit Collaboration. The implicit collaboration mechanisms that we have been developing for software agents in this and related programs is modeled on those of human groups and societies. People share a collective understanding of the world around them, the behavioral guidelines within which they function, and their respective role(s) in that society. Agents exhibit implicit collaboration when they implicitly share information about their application domain, goals of the group, and their respective roles. Examples of mechanisms for implicit collaboration in agent systems include a shared ontology, common communication paradigm, and a shared objective (e.g., to facilitate CSAR operations).

In SARA, agents share a common communication paradigm (the need specification and COM) but do not have a shared ontology. The SARA Dispatcher was not pre-encoded with a shared ontology or schema. Instead it was built it up over time by employing the user to resolve conflicts. While tedious to resolve conflicts initially, subsequent runs of SARA required no user interaction to resolve such conflicts due to persistent storage of the Dispatcher's need-space. Interestingly, this resulted in an ad hoc ontology for the society being created and shared by participating agents.

What the agents in SARA do share are common views of situations. Collaborating agents, like INFAC and a Tech, share a common view of a particular set of things that they are both interested in. Certain objects in the view are the responsibility of INFAC (changing an interaction design to fit a changing domain object), some are the responsibility of a Tech (changing a button state to reflect user input), and each has differing interests, but they jointly manage the ISpace and the objects in it.

An interesting research question is how to share implicit collaboration knowledge with an agent attempting to join a society. This agent does not necessarily share the same knowledge as the other agents, may not know its intended role in that society, and may, in fact, possess knowledge that contradicts the knowledge of other society members. This is analogous to a person joining a new human society (e.g. an employee joining a new company or an immigrant joining a new culture). Some of the responsibility is placed on the new agent (e.g., adopting the communication protocols or ontology) and some is placed on the society (e.g., define the new agent's role and provide feedback to the agent). Finally, the society bears a responsibility to use the unique view of the new agent to modify its collective behavior when appropriate.

Explicit Collaboration. Explicit collaboration allows agents to directly communicate about shared tasks, manage responsibility, and resolve differences. It is through explicit collaboration that agents can learn implicit collaboration. To provide this in an agent

society, dispatcher agents are needed to facilitate both the management of individual and group needs as well as the management of the communication between agents. Such mechanisms exist in human societies in the form of news media, postal services, secretaries, employment agencies, and so on.

The explicit collaboration requirements between SARA agents can be viewed as a conversation, where the state from previous communicative acts and future communicative acts must be taken into account. For example, the Dispatcher might communicate with several agents before finding one or more capable of satisfying a particular need. The state of the conversation (e.g., which agents were asked and which refused) is important to the successful assignment of the need. Research into conversation policies that explicitly define the operational semantics of the agent communication language (e.g., [Bradshaw et al. 1997]) shows promise as a way to retain the state of inter-agent conversations. Future work on the Dispatcher should encapsulate conversation semantics relevant to inter-agent communication, allowing it to transparently maintain the context of the conversation.

10. Bibliography and References

10.1 Technical

J. Ballas, D. McFarlane, L. Achille, J. Stroup, 1996, “Interfaces for Intelligent Control of Data Fusion Processing”, Naval Research Laboratory report NRL/FR/5513-96-9806.

Bradshaw, S. Dutfield, P. Benoit, and J. Woolley, 1997, "KAoS: Toward an Industrial-Strength Open Agent Architecture", in Software Agents, J.M. Bradshaw, editor, AAAI Press, Menlo Park, CA.

A. Cheyer and L. Julia, 1995, "Multimodal Maps: An Agent-Based Approach", International Conference on Cooperative Multimodal Communication, 24-26 May, Eindhoven, The Netherlands.

P. Cohen, A. Cheyer, M. Wang, and S. Baeg, 1994, "An Open Agent Architecture", AAAI Spring Symposium.

P. Cohen and H. Levesque, 1995, "Communicative Actions for Artificial Agents", in Proc. First International Conference on Multi-Agent Systems (ICMAS).

H. Cottam, N. Shadbolt, J. Kingston, H. Beck, and A. Tate, 1995, "Knowledge Level Planning in the Search and Rescue Domain", in Proc. Expert Systems 95 Conference, Cambridge.

K. Decker and V. Lesser, 1995, "Designing a Family of Coordination Algorithms", in Proc. First International Conference on Multi-Agent Systems (ICMAS).

K. Decker, A. Pannu, K. Sycara, and M. Williamson, 1997, "Designing Behaviors for Information Agents", in Proc. First International Conference on Autonomous Agents, February.

R. J. Firby, 1987, "An Investigation into Reactive Planning in Complex Domains", in Proc. National Conf. on Artificial Intelligence, pp. 202-206.

L. Foner, 1997, "Yenta: A Multi-Agent, Referral-Based Matchmaking System", in Proc. First International Conference on Autonomous Agents, February.

M. P. Georgeff and F. F. Ingrand, 1990, "Real-Time Reasoning: The Monitoring and Control of Spacecraft Systems", in Proceedings of the Sixth Conference on Artificial Intelligence Application, pp. 198-204.

J. Gibson, 1979, The Ecological Approach to Visual Perception, Houghton Mifflin.

M. Huber, J. Lee, P. Kenny, E. Durfee, 1996, “UM-PRS V3.0 Programmer and User Guide” University of

Michigan.

D. Kuokka and L. Harada, 1995, "Communication infrastructure for concurrent engineering", Artificial Intelligence for Engineering Design, Analysis, and Manufacturing, vol. 9, pp. 283-297, Cambridge University Press.

J. Lee, M. Huber, E. Durfee, and P. Kenny, 1994, "UM-PRS: An implementation of the Procedural Reasoning System for Multirobot Applications", in Proc. AIAA/NASA Conf. On Intelligent Robots in Field, Factory, Service, and Space, March.

J. McGuire, D. Kuokka, J. Weber, J. Tenenbaum, T. Gruber, and G. Olsen, 1993, "SHADE: Technology for Knowledge-based Collaborative Engineering", in Concurrent Engineering: Research and Applications, vol. 1, pp. 137-146.

M. Minsky, 1985, "Society of the Mind", New York, Simon and Schuster.

K. Nelson, K. Krebsbach, D. Musliner, and R. Penner, 1997, "SARA: The Search and Rescue Assistant", HTC Report #SST-R97-002, January.

K. Nelson, R. Penner, N. Soken, 1998, "SARA: A Multi-Agent System for Collaborative Command and Control", Proc. of the 31st Hawaii International Conference On System Sciences, January.

K. Nelson, R. Penner, N. Soken, K. Krebsbach, and D. Musliner, 1998, "Interaction Society Collaborative Agents System Development Document, CDRL A004, Contract #F30602-95-0177, October.

H. Nii, 1986, "Blackboard systems: The blackboard model of problem solving and the evolution of blackboard architectures", The AI Magazine, V7, n2, Summer.

R. Penner, 1996. “Multi-agent societies for collaborative interaction”, Proceedings of the 40th Annual Meeting of the Human Factors and Ergonomics Society, Philadelphia, PA, September.

R. Penner and K. Nelson, 1997. User Interface Generation Initiative Final Report. Internal Honeywell Technology Center publication.

R. Penner, K. Nelson, N. Soken, 1997, "Facilitating Human Interactions in Mixed Initiative Systems", Proc. IEEE Conference on Systems, Man, and Cybernetics (SMC), October.

D. Riecken, 1997, "The M System" in Software Agents, J. Bradshaw, ed., Cambridge, MIT Press.

J. Searle, 1992, Speech acts: An Essay in the Philosophy of Language, Cambridge University Press, Library of Congress Number 68-24484.

K. Sycara, 1998, “The Many Faces of Agents”, AI Magazine, pp. 11-12, Vol. 19, No. 2, Summer.

L. Suchman, Plans and Situated Actions: The Problem of Human-Machine Communication, Cambridge University Press, 1987.

U.S. Air Force, Force Level Execution Advanced Technology Demonstration Software User’s Manual, 1994, Command, Control, and Communications Division, Rome Laboratory.

10.2 Search and Rescue

Clancy, Tom, 1995, “Fighter Wing: A Guided Tour of an Air Force Combat Wing”, Berkley Books, ISBN: 0-425-14957-9; 1995.

CJCSM 6120.05,1997, Manual For Tactical Command and Control Planning Guidance For Joint Operations, Joint Interface Operational Procedures For Message Text Formats.

Fernandez, D. et al., BDM Federal, 1996, “Joint Search and Rescue Center Information Systems Modernization and Standardization”, prepared for C4I Integration Support Activity.

JIEO Circular 9152, 1995, “Repository of USMTF Program Items for U.S. Implementation Guidance”

Joint Pub 3-50 "National Search and Rescue Manual Volume I: National Search and Rescue System"

Joint Pub 3-50.1 "National Search and Rescue Manual Volume II: Planning Handbook"

Joint Pub. 3-50.2 "Doctrine for Joint Combat Search and Rescue"

Joint Pub 3-50.21 "Joint Tactics, Techniques, and Procedures for Combat Search and Rescue [CSAR]"

Joint Pub. 3-50.3 "Joint Doctrine and Joint Tactics, Techniques and Procedures for Evasion and Recovery"

P. LaValla, R. Stoffel, A. Jones, R. Pargeter, “Search Is An Emergency: A Text for Managing Search Operations”, Fourth Edition, 1995, ISBN: 0-913724-28-9, Emergency Response Institute.

10.3 Standards and Data

T. Carrico, “Command and Control Schema - Revised Draft, Version 0.5.3”, 1996, Object Model Working Group, sponsored by DARPA.

Defense Mapping Agency Mapping, Charting, and Geodesy Utility Software Environment (DMAMUSE) Version 1.1, 1995, Defense Mapping Agency.

Digital Chart of the World, Edition 1: Description and CD-ROM, July 1992, Defense Mapping Agency

DoD HCI Style Guide, 30 September 1992

Graphical Techniques for Force Level Planning, September 1991, Rome Laboratory report RL-TR-01-239,

MIL-HDBK-761A, 30 September 1989, Human Engineering Guidelines for Management Information Systems

MIL-STD-600006, 13 April, 1992, Vector Product Format, Department of Defense.

MIL-STD-2325, 30 September 1994, Common Warfighting Symbology Version 1

MIL-STD-6040, 1997, 1998, USMTF Message Formatting Program

MIL-D-89009, 13 April, 1992, Digital Chart of the World (DCW), Department of Defense.

11. Appendices

The appendices to this document include:

- Appendix 1: Geographic Locations used for testing the SARA MapDemo
- Appendix 2: The classes and hierarchical structure of the CSAR Schema
- Appendix 3: The portion of the CSAR Schema required by DIG.

Appendix 1: Test Locations for the Map Demo

Table of geographic center points to test CD access.

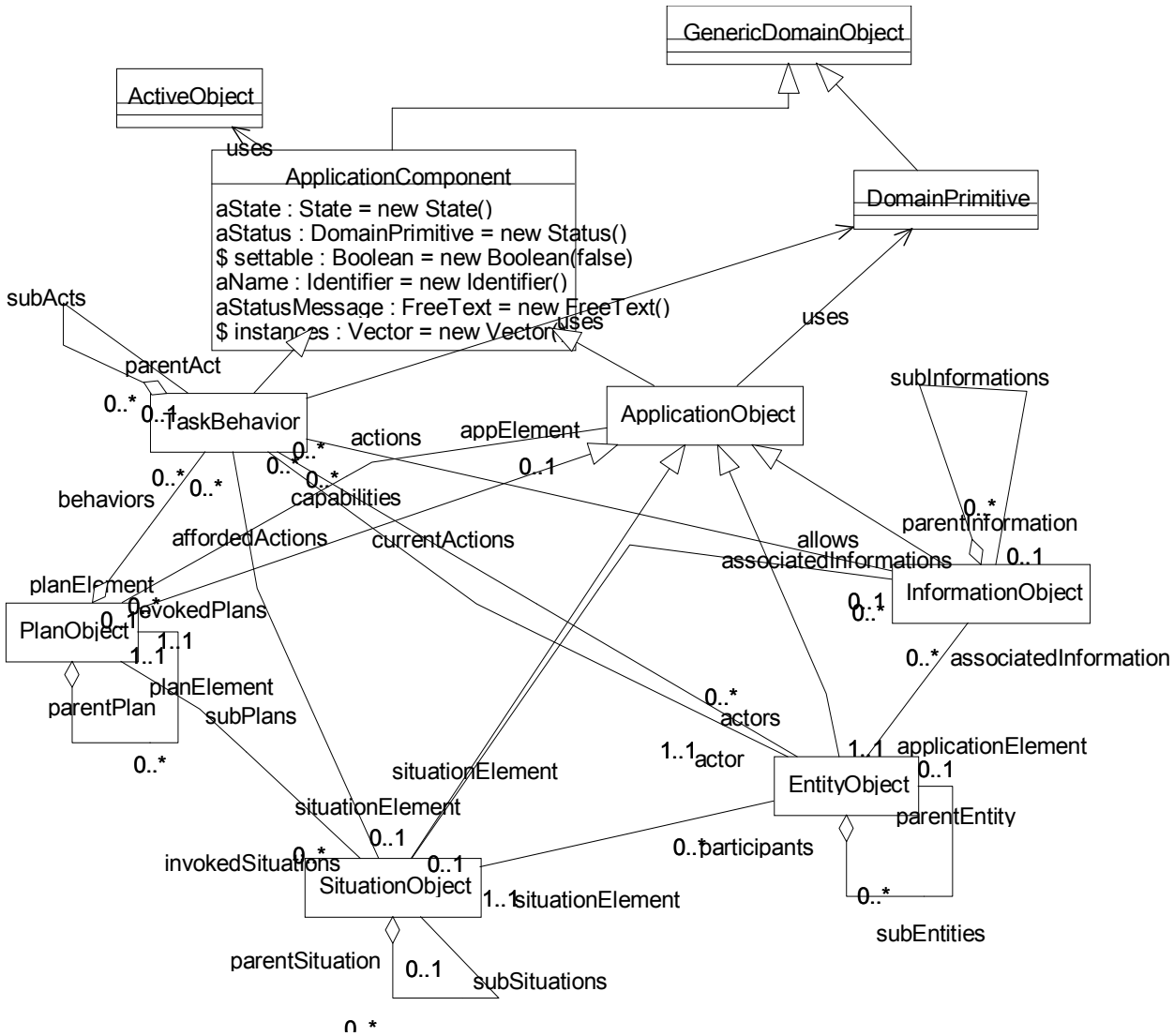
Test	Latitude	Longitude	DCW Libraries
1	40	0	EURNASIA
2	50	-130	NOAMER
3	50	-70	NOAMER
4	-50	-70	SOAMAFR
5	30	50	SASAUS
6	45	135	EURNASIA, SASAUS
7	75	100	EURNASIA
8	35	35	EURNASIA, SASAUS
9	20	-10	SOAMAFR
10	10	100	SASAUS
11	15	-70	NOAMER, SOAMAFR
12	-30	140	SASAUS
13	20	-90	NOAMER
14	40	-10	EURNASIA
15	-10	30	SOAMAFR

Table of place names used to test the gazetteer access.

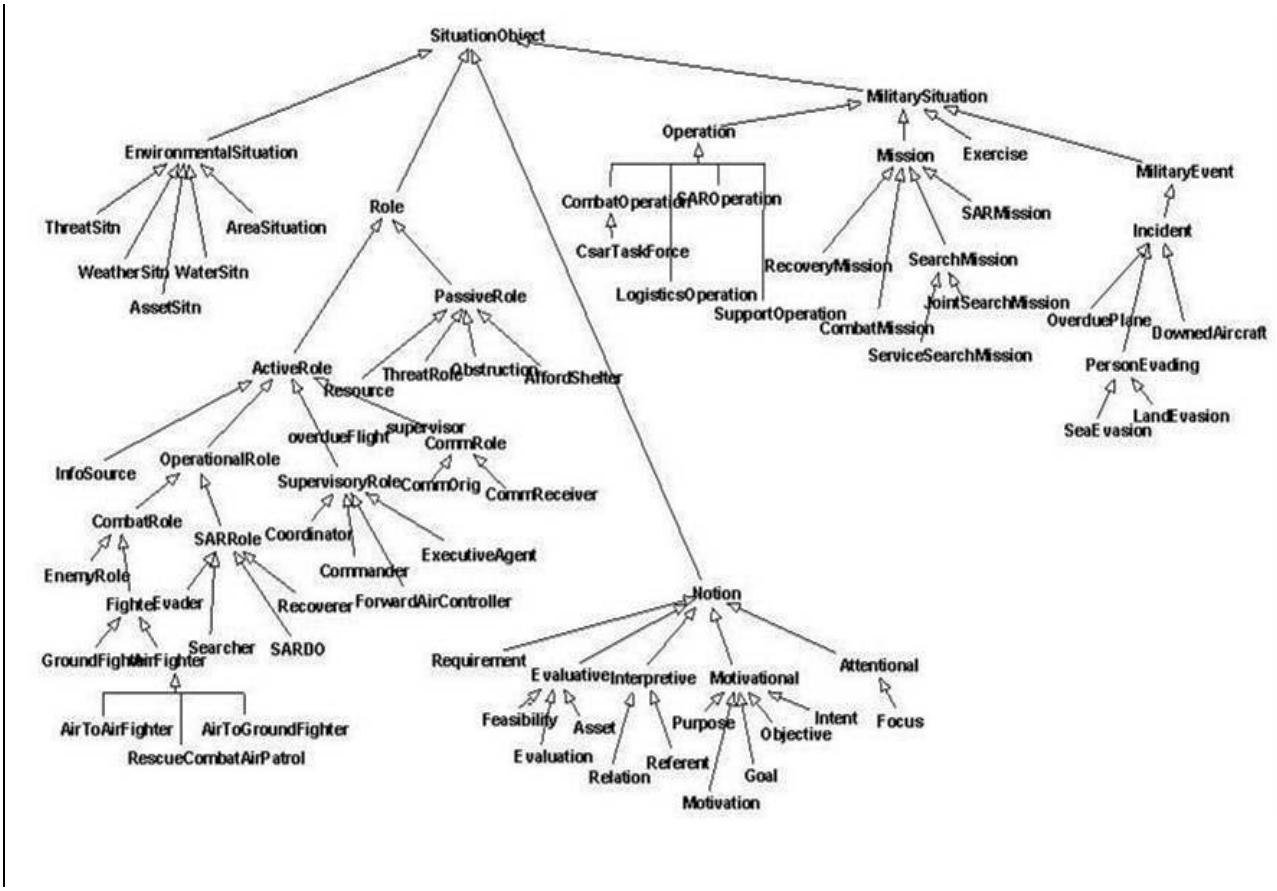
Test	Place name	Library
1	Paris	EURNASIA
2	Turkey	EURNASIA
3	Chicago	NOAMER
4	Seattle	NOAMER
5	Somalia	SOAMAFR
6	Victoria Falls	SOAMAFR
7	Tokyo International	SASAUS
8	Mount Everest	SASAUS

Appendix 2: CSAR Schema Hierarchy

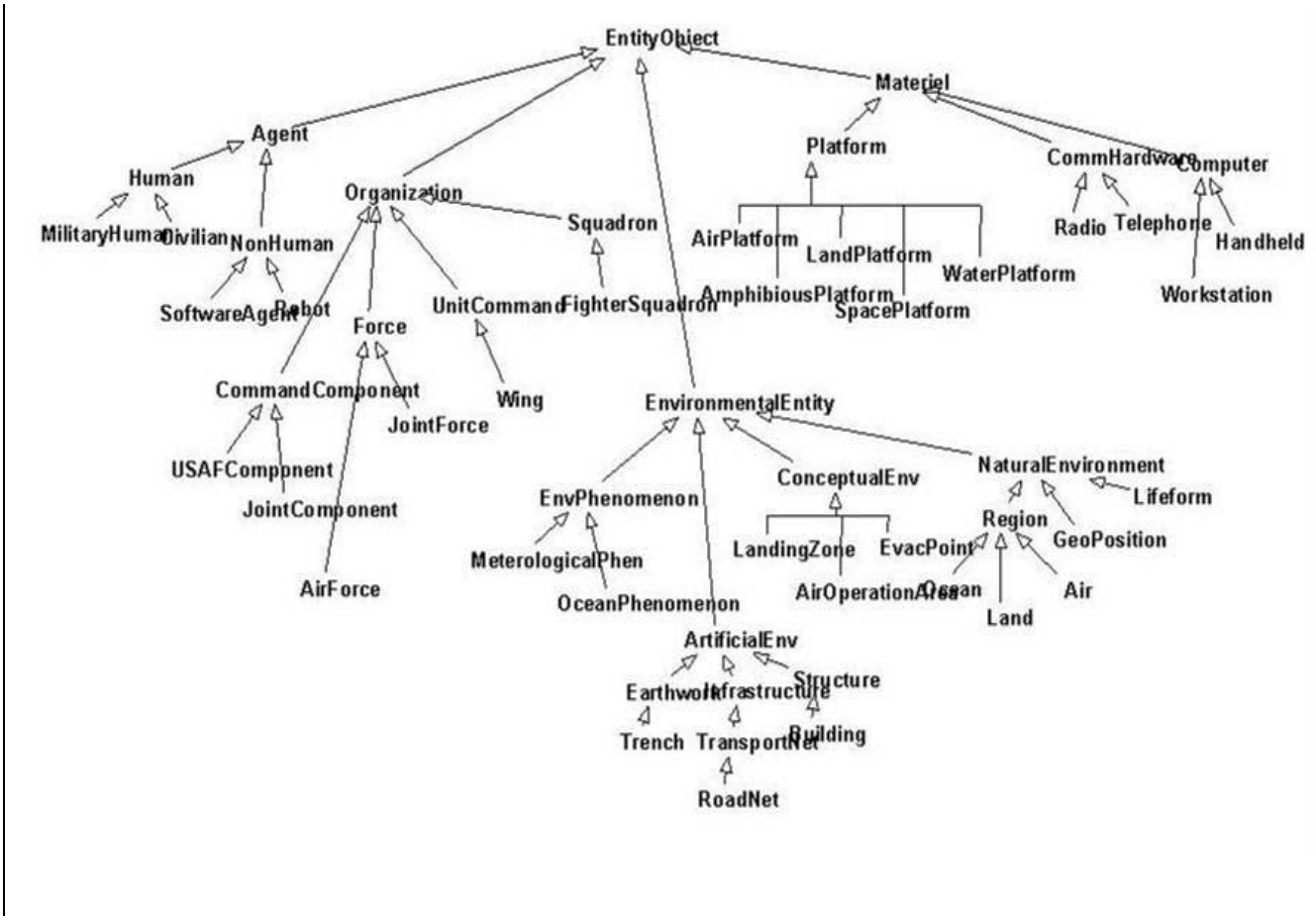
Main schema classes



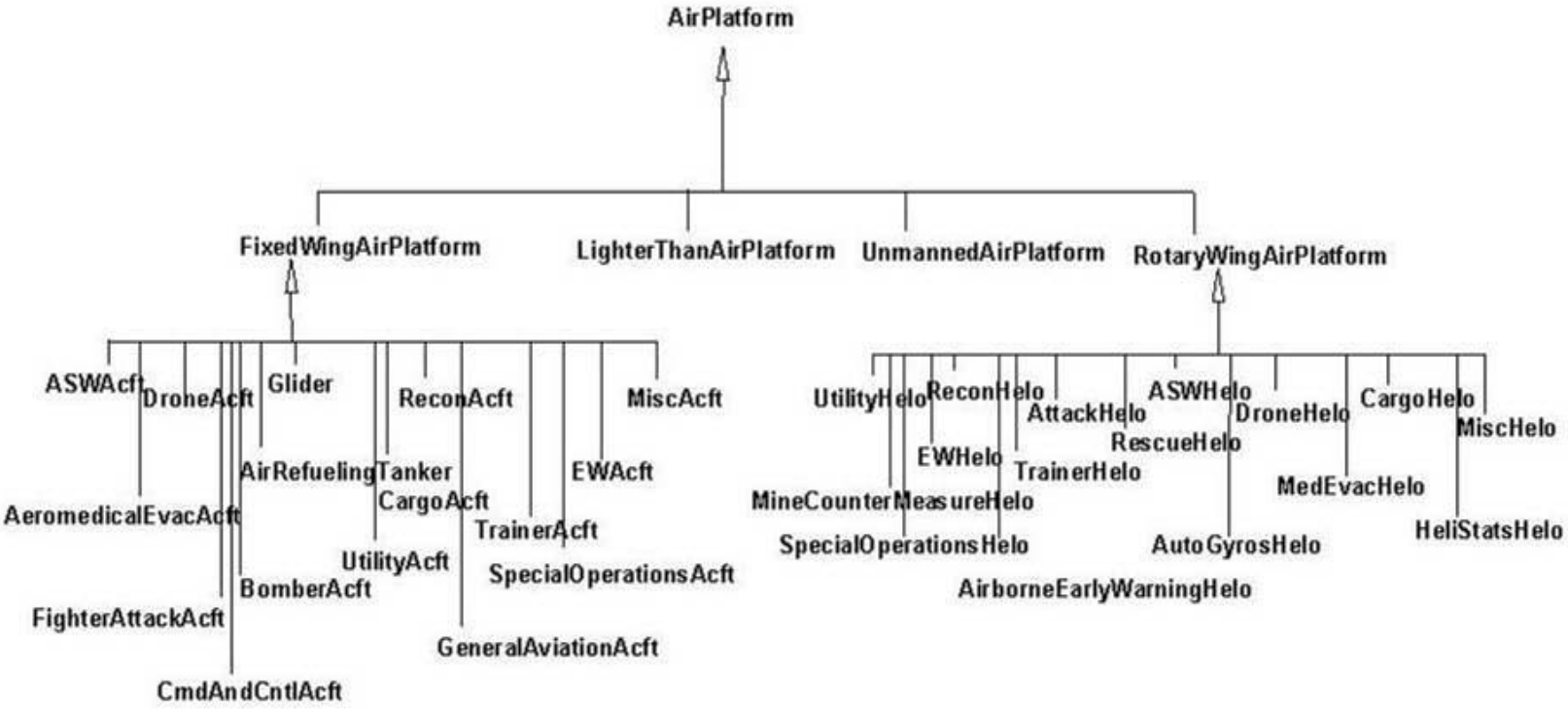
Situation Object Hierarchy

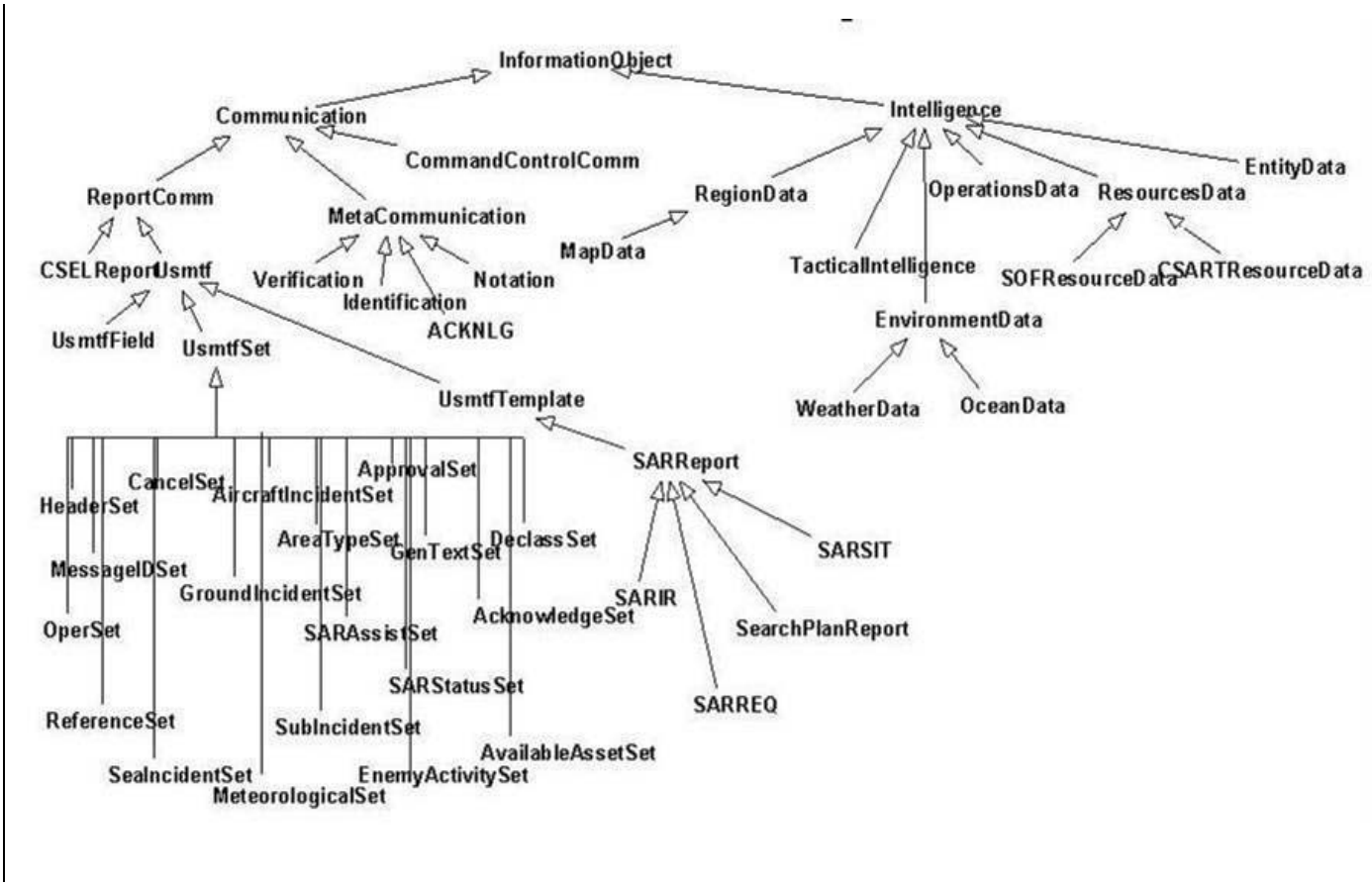


Entity Object Hierarchy

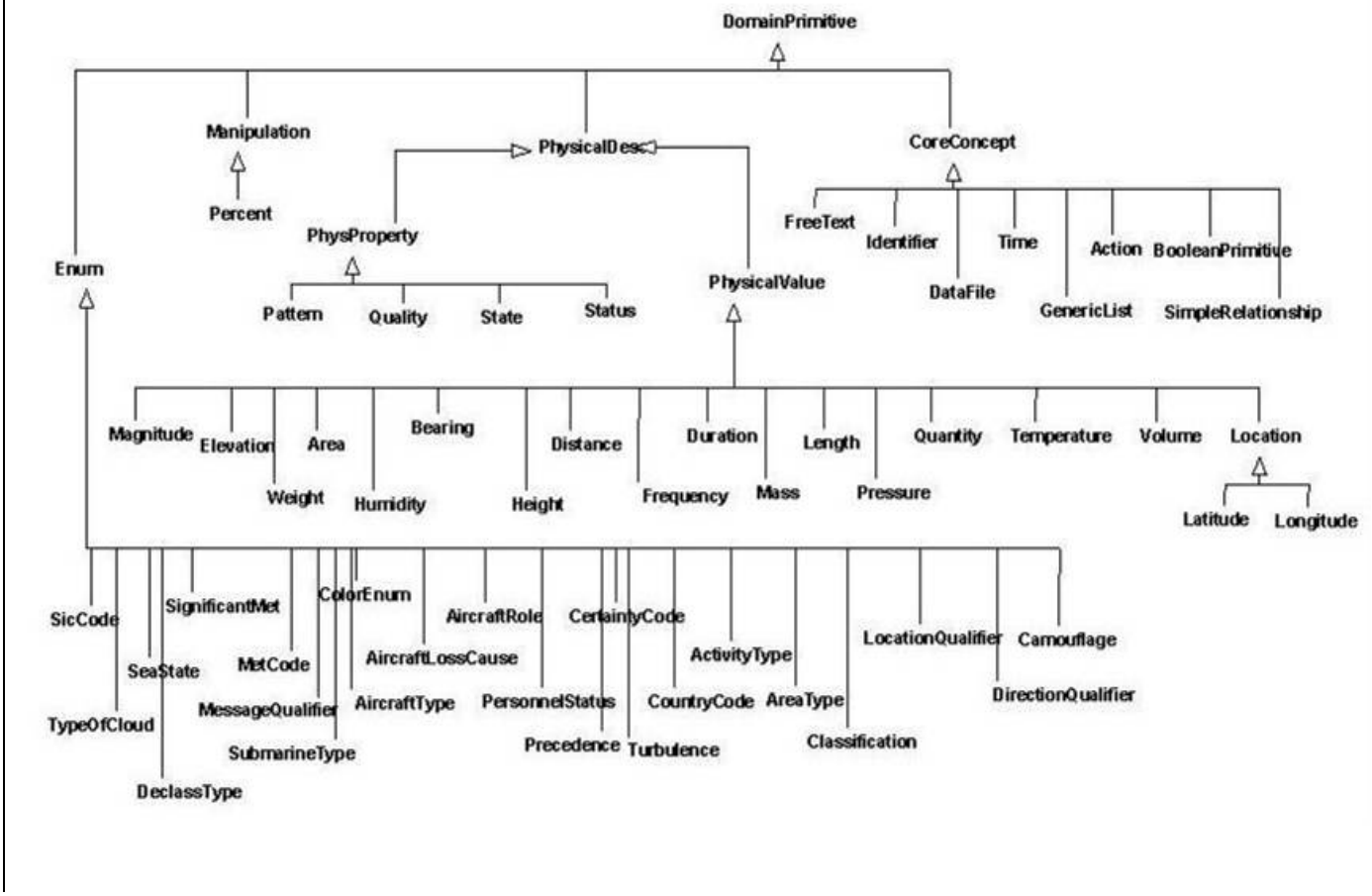


Air Platform Sub-Hierarchy




InformationObject
Communication
Intelligence
CommandControlComm
EntityData
ReportComm
MetaCommunication
RegionData
OperationsData
ResourcesData
MapData
CSELReport
Usmtf
Verification
Identification
Notation
ACKNLG
TacticalIntelligence
SOFResourceData
CSARTResourceData
EnvironmentData
UsmtfField
UsmtfSet
UsmtfTemplate
WeatherData
OceanData
SARReport
CancelSet
ApprovalSet
HeaderSet
AircraftIncidentSet
AreaTypeSet
GenTextSet
DeclassSet
SARSIT
MessageIDSet
GroundIncidentSet
SARIR
AcknowledgeSet
OperSet
SARAssistSet
SearchPlanReport
SARStatusSet
SARREQ
ReferenceSet
SubIncidentSet
AvailableAssetSet
SeaIncidentSet
EnemyActivitySet
MeteorologicalSet

Domain Primitive Hierarchy



Appendix 3:

CSAR Schema Objects Required to Implement DIG